PASSED

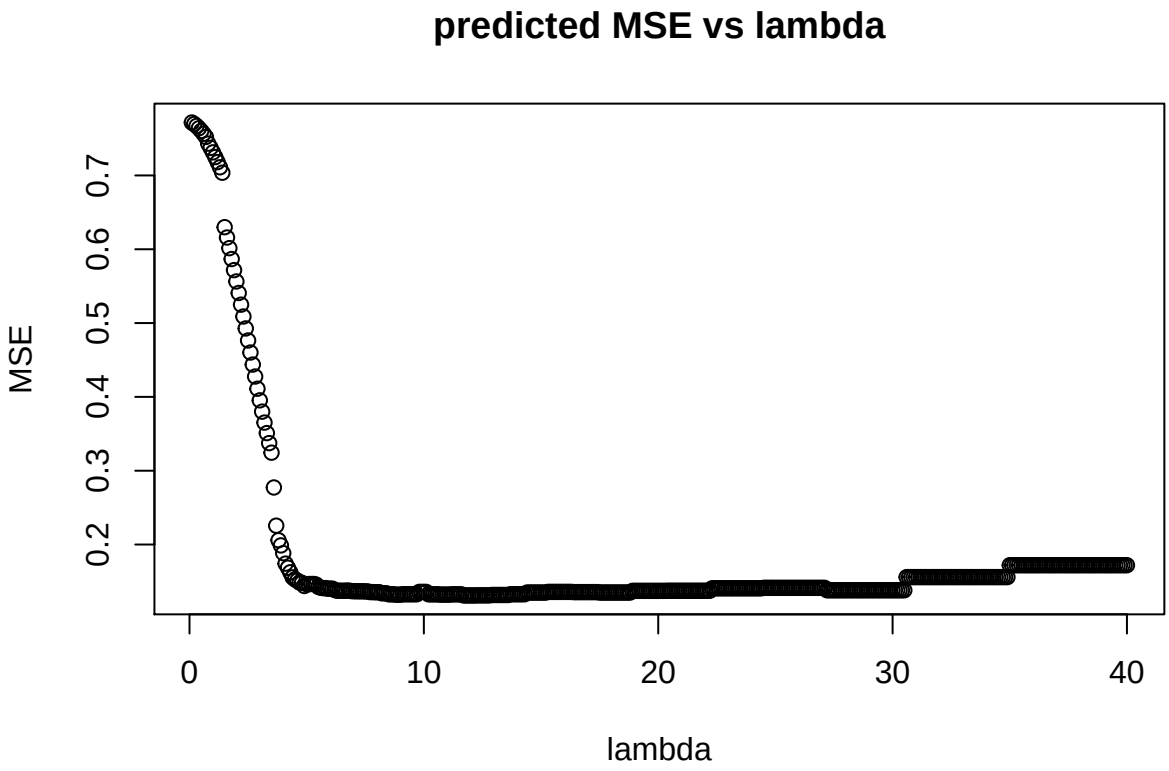
Computer Lab 2

Andrea Bruzzone, Thomas Zhang

2016 M02 9

Assignment 1

Ok, we are going to use the data in `mortality_rate.csv` to fit the log mortality rate of fruit flies versus day of the study. We are going to use the `myMSE` function, which fits using the `loess` function, to find the MSE of the predicted model for some different values of penalty parameter λ. Please see appendix for the code of function `myMSE`.



From the plot, we see that there are likely several values of λ which share the same minimum MSE. we also find that, starting from $\lambda = 0.1$ and progressing by incremental steps of 0.1, it takes 117 `myMSE` evaluations before finding one of the values of λ which produce the minimum MSE. The minimum MSE is 0.131.

Now we use the function `optimize` to find the minimum MSE produced by `myMSE` with an accuracy of 0.01.

```
## $minimum
## [1] 10.69361
##
## $objective
## [1] 0.1321441
```

We find that `optimize` requires 18 `myMSE` evaluations before it finds the value which it recognizes as minimum MSE. This value does not correspond with the one we found previously, possibly because the flattish shape of the curve around the minimum makes it difficult to find the true minimum.

Next we try to find the optimal λ value using `optim` and `method = BFGS`. We set the initial point at $\lambda = 35$.

```
## $par
## [1] 35
##
## $value
## [1] 0.1719996
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
##
## $message
## NULL
```

This way uses 3 `myMSE` evaluations. Referring to the earlier plot, we realize that the quasi-newton method is probably going to fail to find the true minimum MSE, since the gradient at $\lambda = 35$ is zero. Instead, we have found a "local minimum" at $\lambda = 35$. This is a worse result than using `optimize`.

Assignment 2

We write down the log-likelihood function for 100 observations of a normally distributed random variable and find the maximum log-likelihood parameters for μ and σ.

$$L(\mu, \sigma^2, x_1, ..., x_{100}) = (2\pi\sigma^2)^{-100/2} exp(-\frac{1}{2\sigma^2}\sum_{j=1}^{100}(x_j - \mu)^2)$$

$$l(\mu, \sigma^2, x_1, ..., x_{100}) = -\frac{100}{2}ln(2\pi) - \frac{100}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{100}(x_j - \mu)^2)$$

$$\hat{\mu} = \frac{1}{100}\sum_{j=1}^{100} x_j$$

$$\hat{\sigma} = \sqrt{\frac{1}{100}\sum_{j=1}^{100}(x_j - \hat{\mu})^2}$$

```
## [1] "Maximum log-likelihood mean estimator: 1.275528"
```

```
## [1] "Maximum log-likelihood std. deviation estimator: 2.005976"
```

We think it is better to find maximum log-likelihood than maximum likelihood because log-likelihood is a more well behaved function which has a favourable shape for optimization and because the difference in magnitude of function for different parameter values is smaller for log-likelihood.

Now we optimize the negative log-likelihood function with initial parameter values $\mu = 0$ and $\sigma = 1$. We use the `BFGS` method and the conjugate gradient method. For each method, we run optimization with gradient sepcified and without graident specified.

```
## [1] "BFGS method without gradient specified"
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       37       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
## [1] "Conjugate gradient method without gradient specified"
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      180       33
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
## [1] "BFGS method with gradient"
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
```

```
## function gradient
##       38       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
## [1] "Conjugate gradient method with gradient specified"
```

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       53       17
##
## $convergence
## [1] 0
##
## $message
## NULL
```

All the final parameter values are the same, and all algorithms converge. For `BFGS` method, it makes no difference whatsoever whether the gradient is provided or not. For the conjugate gradient method, the number of iterations required decrease when the gradient is provided. We would recommend using the BFGS method with gradient provided if possible, because it requires the smallest number of iterations.

Contributions

Both Group memebers contributed with their code and their ideas. Andrea contributed the code for assignment 2 and Thomas contributed the code for assignment 1. We also discussed together the questions and how to interpret results.

Appendix

R code

```
data <- read.csv2("mortality_rate.csv")
#head(data)
data[,3] <- log(data$Rate)
names(data)[3] <- "LMR"
n = dim(data)[1]
set.seed(123456)
id = sample(1:n, floor(n * 0.5))
train = data[id,]
```

```
test = data[-id,]

myMSE <- function(lambda,pars){
  lel <- unlist(pars)
  frame <- data.frame(X = lel[1:68],Y = lel[69:136], Xtest = lel[137:204], Ytest = lel[205:272])
  model1 <- loess(Y ~ X, data = frame, enp.target = lambda)
  pred <- predict(model1, newdata = frame[,3])
  #print(pred)
  predMSE <- mean((pred - frame[,4])^2)
  #print(c("The MSE of loess fit is: ",signif(predMSE,4)))
  return(predMSE)
}
pars <- list(X=train$Day,Y=train$LMR,Xtest=test$Day,Ytest=test$LMR)
res <-myMSE(0.1,pars)

lambvec <- seq(from=0.1,to=40,by=0.1)
resvec <- c()
for(i in 1:length(lambvec)){
  resvec <- c(resvec,myMSE(lambvec[i],pars))
}
plot(lambvec,resvec,main="predicted MSE vs lambda",xlab ="lambda",ylab = "MSE")
op1<-optimize(myMSE,c(0.1,40),pars=pars, tol= 0.01) # 18 iterations required
op1
op2 <- optim(35,myMSE,method = "BFGS",pars=pars) # 3 iterations but it is worse value
op2
load("data.RData")
muhat <- sum(data)/100
paste("Maximum log-likelihood mean estimator:",round(muhat,6))
sigmasquarehat <- sum((data - muhat)^2) / 100
sigmahat <- sqrt(sigmasquarehat)
paste("Maximum log-likelihood std. deviation estimator:",round(sigmahat,6))


#2.4
minusloglike <- function(pars){
  mu <- pars[1]
  sig <- pars[2]
  loglike <- - (-50*log(2*pi) - 50*log(sig^2) - 1/(2*sig^2)*sum((data- mu)^2))
  return(loglike)
}

gradient <- function(pars){
  mu <- pars[1]
  sig <- pars[2]
  c(-(1/sig^2*sum(data - mu)), -(-100/sig + 1/sig^3*sum((data- mu)^2)))
}


oplog <- optim(c(0,1), fn = minusloglike, method = "BFGS")
paste("BFGS method without gradient specified")
oplog

oplog2 <- optim(c(0,1), minusloglike, method = "CG")
```

```r
paste("Conjugate gradient method without gradient specified")
oplog2

#specified the gradient
oploggrad <- optim(c(0,1), fn = minusloglike, method = "BFGS", gr = gradient)
paste("BFGS method with gradient")
oploggrad

oplog2grad <- optim(c(0,1), minusloglike, method = "CG", gr = gradient)
paste("Conjugate gradient method with gradient specified")
oplog2grad
## NA
```